Submission Worksheet

CLICK TO GRADE

https://learn.ethereallab.app/assignment/IT114-003-F2024/it114-module-5-project-milestone-1/grade/vvh

Course: IT114-003-F2024
Assigment: [IT114] Module 5 Project Milestone 1
Student: Valeria C. (vvh)

Submissions:

Submission Selection

1 Submission [submitted] 10/17/2024 7:18:02 PM

Instructions

COLLAPSE

Overview Video: https://youtu.be/A2yDMS9TS1o

1. Create a new branch called Milestone1
2. At the root of your repository create a folder called Project if one doesn't exist yet
    1. You will be updating this folder with new code as you do milestones
    2. You won't be creating separate folders for milestones; milestones are just branches
3. Copy in the code from Sockets Part 5 into the Project folder (just the files)
    2. https://github.com/MattToegel/IT114/tree/M24-Sockets-Part5
4. Fix the package references at the top of each file (these are the only edits you should do at this point)
5. Git add/commit the baseline and push it to github
6. Create a pull request from Milestone1 to main (don't complete/merge it yet, just have it in open status)
7. Ensure the sample is working and fill in the below deliverables 1. Note: Don't forget the client commands are /name and /connect
8. Generate the output file once done and add it to your local repository
9. Git add/commit/push all changes
10. Complete the pull request merge from the step in the beginning
11. Locally checkout main
12. git pull origin main

Branch name: Milestone1

Group

100%

Group: Start Up
Tasks: 2
Points: 3

COLLAPSE

Task

100%

Group: Start Up
Task #1: Start Up
Weight: ~50%
Points: ~1.50

COLLAPSE

Details:

Important: Code screenshots should be fairly concise (try to show only the sections of code relevant to the question)

Capturing all possible code (i.e., including a lot of irrelevant code) can lead to a reduced grade.

Columns: 4

Sub-Task 100% Group: Start Up Task #1: Start Up Sub Task #1: Show the Server

Task Screenshots

Gallery Style: 2 Columns

4 2 1

server starting via command line, listening for connections screenshot

Caption(s) (required) ✓

Caption Hint: *Describe/highlight what's*

Sub-Task 100% Group: Start Up Task #1: Start Up Sub Task #2: Show the Server

Task Screenshots

Gallery Style: 2 Columns

4 2 1

server code listens for connections

Caption(s) (required) ✓

Caption Hint: *Describe/highlight what's being shown (ucid/date must be present)*

Task

Sub-Task 100% Group: Start Up Task #1: Start Up Sub Task #3: Show the Client

Task Screenshots

Gallery Style: 2 Columns

4 2 1

client starting via command line screenshot

Caption(s) (required) ✓

Caption Hint: *Describe/highlight what's being shown*

Sub-Task 100% Group: Start Up Task #1: Start Up Sub Task #4: Show the Client

Task Screenshots

Gallery Style: 2 Columns

4 2 1

client code that waits for user input screenshot

client code that prepares client screenshot

Caption(s) (required) ✓

Caption Hint: *Describe/highlight what's being shown (ucid/date must be present)*

being shown

Response Prompt

Briefly explain the code related to starting up and waiting for connections

Response:

The server starts by printing a server starting to indicate that it has begun. it then listens on a 3000 port to listen for new clients connections. then followed by the waiting for next client line showing that the server is actively waiting for a client to connect and when the client connects, the server accepts the connection and then starts a new thread to handle that client. the server keeps going over the loop waiting for more clients to connect while handling each one in a separated thread.

Task Response Prompt

Briefly explain the code/logic/flow leading up to and including waiting for user input

Response:

In this part of the code, the client code prepares the client by connecting to the server using a socket and setting the communication channels to send and receive the data. the client runs a separste thread to listen for messages from the server while it waits for user input by commands like /name or /connect and when the user enters a command, it processes it to either connect to the server or set the client name and if the client is connected to the server it sends the user's messages, otherwise it prompts the user connect first.

End of Task 1

Task

100%

Group: Start Up
Task #2: Connecting
Weight: ~50%
Points: ~1.50

COLLAPSE

Details:

Important: Code screenshots should be fairly concise (try to show only the sections of code relevant to the question)

Capturing all possible code (i.e., including a lot of irrelevant code) can lead to a reduced grade.

Columns: 2

Sub-Task

100%

Group: Start Up
Task #2: Connecting
Sub Task #1: Show 3 Clients connecting to the Server

Task Screenshots

Gallery Style: 2 Columns

4 2 1

3 clients connecting to server screenshot

Caption(s) (required) ✓

Caption Hint: *Describe/highlight what's being shown*

Sub-Task

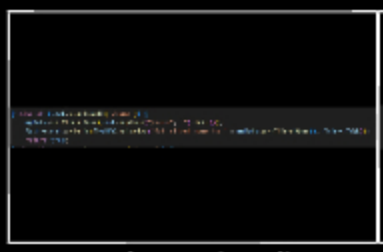
100%

Group: Start Up
Task #2: Connecting
Sub Task #2: Show the code related to Clients connecting to the Server (including the two needed commands)

Task Screenshots

Gallery Style: 2 Columns

4 2 1



command 1: setting client name screenshot

Command 2 connecting to the server

connect method in client.java

listening for client connections in server.java

Caption(s) (required) ✓

Caption Hint: *Describe/highlight what's being shown (ucid/date must be present)*

Task Response Prompt

Briefly explain the code/logic/flow

Response:

The process starts with the client setting a name using the /name command. After the name is set, the client connects to the server with the /connect command, which includes the server's address and port. The server listens for incoming connections using the serversocket, and when a client connects, it creates a serverthread to handle each client

End of Task 2

End of Group: Start Up
Task Status: 2/2

Group

100%

Group: Communication
Tasks: 2
Points: 3

COLLAPSE

Task

100%

Group: Communication
Task #1: Communication
Weight: ~50%
Points: ~1.50

COLLAPSE

Details:
Important: Code screenshots should be fairly concise (try to show only the sections of code relevant to the question)

Capturing all possible code (i.e., including a lot of irrelevant code) can lead to a reduced grade.

Columns: 4

Sub-Task 100% Group: Communicatio Task #1: Communicatio Sub Task #1: Show each Client

Sub-Task 100% Group: Communicatio Task #1: Communicatio Sub Task #2: Show the code

Sub-Task 100% Group: Communicatio Task #1: Communicatio Sub Task #3: Show the code

Sub-Task 100% Group: Communicatio Task #1: Communicatio Sub Task #4: Show the code

Task Screenshots

Gallery Style: 2 Columns

4 2 1

client sending and receiving messages screenshot

Caption(s) (required) ✓
Caption Hint:
Describe/highlight what's being shown

Task Screenshots

Gallery Style: 2 Columns

4 2 1

code related to the client-side of getting a user message screenshot

sending message over the socket screenshot

Caption(s) (required) ✓
Caption Hint:
Describe/highlight what's being shown (avoid/date must Describe/highlight what's

Task Screenshots

Gallery Style: 2 Columns

4 2 1

codes receive message from a client and relays it to other clients in the room screenshot

Caption(s) (required) ✓
Caption Hint:

Task Screenshots

Gallery Style: 2 Columns

4 2 1

code related to the client receiving messages from the server-side in client.java

display message code

Caption(s) (required) ✓
Caption Hint:
Describe/highlight what's

being shown (ucid/date must be present)

Task Response Prompt

Briefly explain the code/logic/flow involved

Response:

The client waits for user input through the console using the listentoinput method. When the user types a message, the client first checks if the input is a command like /name or /connect. If it's not a command, the input is treated as a regular message. Then, the client wraps this message in a payload object and sends it to the server using the sendmessage method. The message is then transmitted over the socket connection, allowing the server to receive and process it.

Describe/highlight what's being shown (ucid/date must be present)

Task Response Prompt

Briefly explain the code/logic/flow involved

Response:

On the server side, the server continuously listens for incoming messages from connected clients through the processpayload method in the serverthread class. When a message is received, the server identifies the payload as a message and uses the sendmessage method in the room class to broadcast the message to all other clients in the same room. Each client in the room will then receive the message so all connected clients have communication,

Describe/highlight what's being shown (ucid/date must be present)

Task Response Prompt

Briefly explain the code/logic/flow involved

Response:

The client listens for messages from the server in the listentoserver method. When a message is received, it arrives as a payload object from the server. The client processes the message and displays it on the console using the processpayload method. This allows the user to see messages from other clients

End of Task 1

Task

100%

Group: Communication
Task #2: Rooms
Weight: ~50%
Points: ~1.50

COLLAPSE

Details:

Important: Code screenshots should be fairly concise (try to show only the sections of code relevant to the question)

Capturing all possible code (i.e., including a lot of irrelevant code) can lead to a reduced grade.

Columns: 4

Sub-Task 100% Group: Communicatio Task #2: Rooms Sub Task #1: Show Clients can

Task Screenshots

Gallery Style: 2 Columns

4 2 1

clients can create rooms screenshot

Caption(s) (required) ✓

Caption Hint:

Describe/highlight what's being shown

Sub-Task 100% Group: Communicatio Task #2: Rooms Sub Task #2: Show Clients can

Task Screenshots

Gallery Style: 2 Columns

4 2 1

leave/join message between rooms screenshot

Caption(s) (required) ✓

Caption Hint:

Describe/highlight what's being shown

Sub-Task 100% Group: Communicatio Task #2: Rooms Sub Task #3: Show the Client

Task Screenshots

Gallery Style: 2 Columns

4 2 1

command to create room

command to joinroom

Caption(s) (required) ✓

Caption Hint:

Describe/highlight what's being shown (ucid/date must be present)

Task Response Prompt

Briefly explain the code/logic/flow involved

Response:

The client side code allows users to create or join rooms by typing commands. When a user types /createroom (roomname), the client extracts the room name, wraps it in a payload object, and sends it to the server as a request to create the room. the same condition happens when a user types /joinroom (roomname), the client sends a payload to the server to join an existing room. Both commands are

Sub-Task 100% Group: Communicatio Task #2: Rooms Sub Task #4: Show the

Task Screenshots

Gallery Style: 2 Columns

4 2 1

ServerThread Code handling create/join room requests

room code handling the room creation and joining

Caption(s) (required) ✓

Caption Hint:

Describe/highlight what's being shown (ucid/date must be present)

Task Response Prompt

Briefly explain the code/logic/flow involved

Response:

The serverthread and room classes handle the create/join process when a client sends a request to create a room, the serverthread processes it by calling the handlecreateroom method in the room class. This method checks if the room already exists using the server class, and if it doesn't, the room is created and the client is automatically added to it.

processed in the processclientcommand method, and the actual data is sent to the server using the send method. The server receives these requests and processes them, creating or adding the client to the specified room

For joining a room, the serverthread calls the handlejoinroom method, which checks if the room exists and adds the client to it if it does. The room class manages the clients within each room, making sure that when a client joins or leaves, the server and other clients in the room are updated accordingly

Sub-Task 100% Group: Communicatio Task #2: Rooms Sub Task #5: Show the Server

Task Screenshots

Gallery Style: 2 Columns

4 2 1

server code managing rooms

Caption(s) (required) ✓

Caption Hint: *Describe/highlight what's being shown (ucid/date must be present)*

Task Response Prompt

Briefly explain the code/logic/flow involved

Response:

When a client sends a

Sub-Task 100% Group: Communicatio Task #2: Rooms Sub Task #6: Show that Client

Task Screenshots

Gallery Style: 2 Columns

4 2 1

showing messages are constrained screenshot

Caption(s) (required) ✓

Caption Hint: *Describe/highlight what's being shown*

Task Response Prompt

Briefly explain why/how it works this way

Response:

The server class ensures

request to create or join a room, the serverthread processes the request using its processpayload method. If the request is to create a room, the room class's handlecreateroom() method is called, which checks with the Server class's createroom method to see if the room already exists. If not, the room is created and the client is added to it. If the request is to join a room, the handlejoinroom method is called, and the Server class's joinroom method adds the client to the requested room if it exists. The Server manages all rooms globally, making sure that the rooms are created and worked properly and clients are assigned to the correct rooms

that client messages are constrained to their specific rooms by managing clients within the room class. Each room has a list of clients connected to it, and when a client sends a message, the room class's sendMessage method is called. This method loops through the list of clients in that particular room and sends the message only to those clients. If clients are in different rooms, they are stored in separate lists within their respective rooms, so messages from one room do not get sent to clients in another room which prevents cross-room messaging

End of Task 2

End of Group: Communication
Task Status: 2/2

Group

100%

Group: Disconnecting/Termination
Tasks: 1
Points: 3

COLLAPSE

Task

100%

Group: Disconnecting/Termination
Task #1: Disconnecting
Weight: ~100%
Points: ~3.00

COLLAPSE

Details:

Important: Code screenshots should be fairly concise (try to show only the sections of code relevant to the question)

Capturing all possible code (i.e., including a lot of irrelevant code) can lead to a reduced grade.

Columns: 4

Sub-Task 100% Group: Disconnecting/ Task #1: Disconnecting Sub Task #1: Show Clients

Task Screenshots

Gallery Style: 2 Columns

4 2 1

clients gracefully disconnecting screenshot

Caption(s) (required) ✓

Caption Hint:

Describe/highlight what's being shown

Sub-Task 100% Group: Disconnecting/ Task #1: Disconnecting Sub Task #2: Show the code

Task Screenshots

Gallery Style: 2 Columns

4 2 1

close server connection

close the connection

command to disconnect

Caption(s) (required) ✓

Caption Hint:

Describe/highlight what's being shown (ucid/date must be present)

Task Response Prompt

Briefly explain the code/logic/flow involved

Response:

Client sends a disconnect command, /disconnect.

Sub-Task 100% Group: Disconnecting/ Task #1: Disconnecting Sub Task #3: Show the Server

Task Screenshots

Gallery Style: 2 Columns

4 2 1

server terminating screenshot

Caption(s) (required) ✓

Caption Hint:

Describe/highlight what's being shown

Sub-Task 100% Group: Disconnecting/ Task #1: Disconnecting Sub Task #4: Show the Server

Task Screenshots

Gallery Style: 2 Columns

4 2 1

method shutdown ensures all clients are disconnected and resources are cleaned up before the server shuts down

shutdown method iterates through all rooms ensuring that server terminates by disconnecting clients and cleaning up resources

Caption(s) (required) ✓

Caption Hint:

Describe/highlight what's being shown (ucid/date must be present)

Task Response Prompt

Briefly explain the code/logic/flow involved

Response:

The client creates a disconnect payload and sends it to the server. On the server side, the ServerThread receives this payload and calls the disconnect method in the room class, which removes the client from the room, notifies other clients in the room, and clears the client's connection. The server then invokes the cleanup method to ensure that all resources associated with the client, such as the input/output streams and the socket, are properly closed

Response:

The server handles termination by using a JVM shutdown hook and the shutdown method to ensure a smooth shutdown. When the server receives a termination signal, the shutdown hook triggers the shutdown method. This method iterates over all active rooms and calls disconnectAll() on each room, ensuring that all clients are disconnected before the server fully shuts down

End of Task 1

End of Group: Disconnecting/Termination
Task Status: 1/1

Group

100%

Group: Misc
Tasks: 3
Points: 1

COLLAPSE

Task

100%

Group: Misc
Task #1: Add the pull request link for this branch
Weight: ~33%
Points: ~0.33

COLLAPSE

Task URLs

URL #1
https://github.com/vvh24/vvh-IT114-003/pull/11

URL
https://github.com/vvh24/vvh-IT114-003/pull/11

End of Task 1

Task

100%

Group: Misc
Task #2: Talk about any issues or learnings during this assignment
Weight: ~33%
Points: ~0.33

COLLAPSE

Details:
Few related sentences about the Project/sockets topics

Task Response Prompt

Response:

I didn't run into any issues. At the beginning for some reason I struggle passing the files from part5 to my local even though I used vscode as usual but I was getting errors and specially when I tried to compile the files. I had to delete and go over each of them again until I finally could solve it and everything worked. then, I went over the instructions step by step and familiarize myself with the code provided as template for our chosen project later of milestone 2. Regarding sockets, this is a new concept for me, but it is very interesting to see how the client-server code and the related code and how it works through files and also being able to test it out.

End of Task 2

Task

100%

Group: Misc
Task #3: WakaTime Screenshot
Weight: ~33%
Points: ~0.33

COLLAPSE

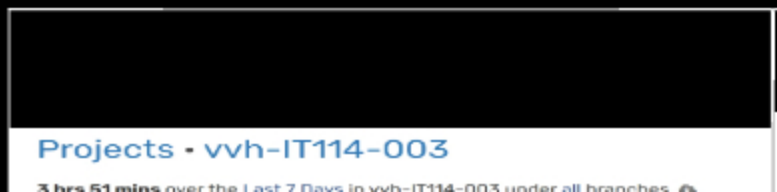
Details:
Grab a snippet showing the approximate time involved that clearly shows your repository.

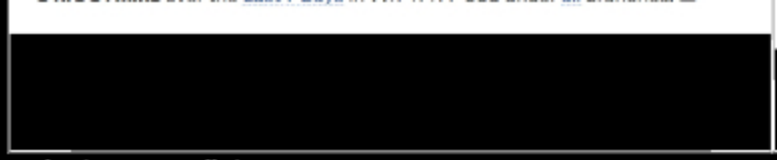
The duration isn't considered for grading, but there should be some time involved.

Task Screenshots

Gallery Style: 2 Columns

4 2 1

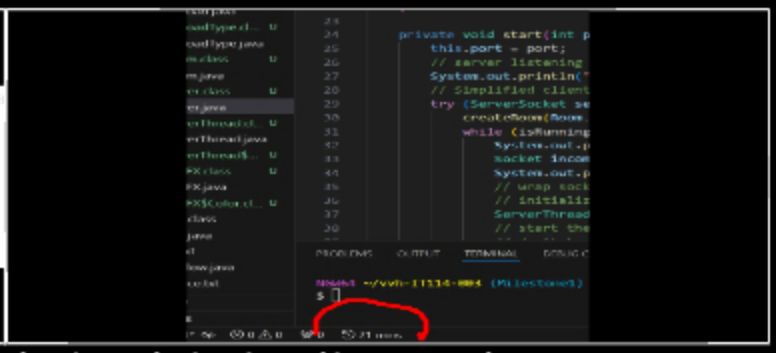




wakatime overall time

time spent in each file

time spent the day before when I started working on milestone1



showing wakatime is working on vscode

End of Task 3

End of Group: Misc
Task Status: 3/3

End of Assignment